HT32 系列单片机 PDMA 说明

文件编码：AN0310S

简介

关于PDMA

部分的 HT32 系列单片机，包含 HT32F1755/1765/2755 等型号在内，内建了 8~12 个 channel 的 PDMA，支持内部 FLASH、SRAM、USB RAM 和 peripheral 之间大量或连续性数据的转移。每个 PDMA channel 的设置都是独立的，并支持特定 peripheral 的单方向 request 和软件触发，多个 PDMA channel 的传输可以一同进行，但依照所设置的优先级高低，交错使用总线的传输带宽。

数据传输

PDMA channel 的数据传输，是由许多 block 所组成的，每个 block 的数据量大小，是由 block length 和 data width 所决定。Block count 和 block length 最大支持到 65,535，当 block length 被设成 4 的倍数时，可以获得最佳的传输效率；data width 可选择 8-bit、16-bit 或 32-bit。如果使用 PDMA 来做大量数据的转移时，例如用软件触发来做 memory copy 等操作，应该将 block length 设小，block count 设大，优先级设低，来避免大区块的数据转移造成其它 peripheral 的 buffer 或 FIFO 发生 overflow/underflow。

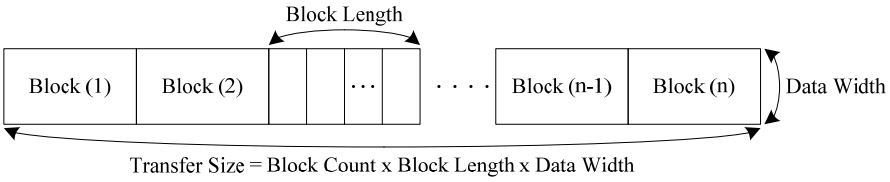

图 1　数据传输

中断控制

PDMA channel 的 interrupt status 分为 BE (Block End)、HT (Half Transfer)、TC (Transfer Complete) 和 TE (Transfer Error) 四种，以及一个控制用的 GE (Global Event)。其中 BE 代表每个 block 传输完成，HT 代表整个数据传输完成一半，TC 代表整个数据传输完成，TE 代表数据传输发生错误。要产生 BE/HT/TC/TE 的 interrupt status，除了要开启 BE/HT/TC/TE 的 interrupt enable，还必须开启 GE 的 interrupt enable，请参考下图。

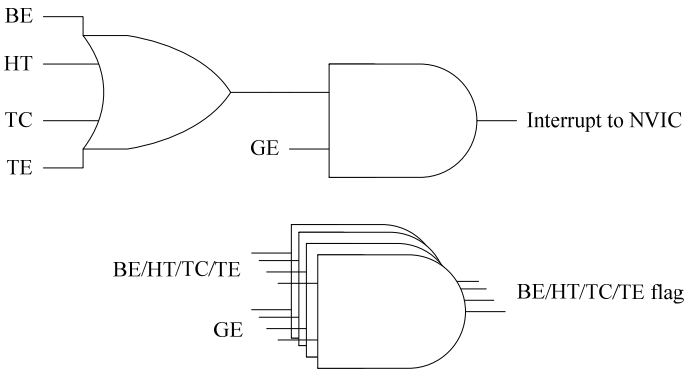

图 2　中断状态控制

例如：有笔 96-byte 的数据要 PDMA 传输，设置时将其分成 3 个 block，且 BE/HT/TC 和 GE 的 interrupt enable 都开启：

- 第一个 block 传输完成后，BE 和 GE 的 interrupt status 会被设为 1，清除 BE 的 interrupt status，会连同 GE 的 interrupt status 一起清除
- 第二个 block 传输完成后，除了 BE 和 GE 之外，HT 的 interrupt status 也会被设为 1，清除 BE 的 interrupt status，GE 的 interrupt status 也会被清除，但 HT 的 interrupt status 不会被清除
- 第三个 block 传输完成后，BE、GE 和 TC 的 interrupt status 都会被设为 1，清除 GE 的 interrupt status，则包含 BE、TC 以及 HT (第二个 block 传输完成所产生)的 interrupt status 都会一起清除

TE 的 interrupt status 只在以下情况才会被设为 1：

- 当 block count 或 block length 为 0，却又收到 peripheral request 或是软件触发
- 来源或是目的地址不存在(reserved)
- 错误的总线行为

地址模式

PDMA 的来源和目的地址分为 linear、circular 和 fixed 三种模式，可分别设置用来支持不同的 peripheral 和应用，其中 linear 和 circular 地址模式，还可选择地址递增或是递减。使用 Linear 和 circular 地址模式，在完成一笔数据的传输后，除了会将 CBLKLEN (current block length) 减 1 外，还会依照 data width 的设置，递增或是递减 CSADR (current source address) 和 CDADR (current destination address)，两者的差异在每个 block 传输完成后，使用 circular 地址模式的 CSADR 或 CDADR，会 reload 成 SADR (source address) 或 DADR (destination address) 的值，而 linear 地址模式则是继续递增或递减下去。使用 Fixed 地址模式，则 CSADR 或 CDADR 不会改变。

Source Address Mode	Desination Address Mode
Linear Increment/Decrement	Linear Increment/Decrement
Linear Increment/Decrement	Circular Increment/Decrement
Linear Increment/Decrement	Fixed
Circular Increment/Decrement	Linear Increment/Decrement
Circular Increment/Decrement	Circular Increment/Decrement
Fixed	Linear Increment/Decrement
Fixed	Fixed

表 1　支持的地址模式组合

注意事项

因为设计上的限制，所以使用 PDMA 读取 FLASH 数据时，有 64KB boundry 的问题存在。例如来源地址 SADR 设为 0x0000_FE00，传输大小为 1 KB，当传输进行到第 513-byte 时，因为地址刚好跨越 64 KB boundary，所以 PDMA 会读到 FLASH 地址 0x0000_0000 的值，而不是地址 0x0001_0000 的值。要解决这个问题，必需将传输分成两次进行，第一次传输的 SADR 设为 0x0000_FE00，传输大小为 512-byte，第二次传输的 SADR 设为 0x0001_0000，传输大小为 512-byte。

范例

使用Linear Increment与Fixed地址模式

以 SPI TX 为例，假设 SRAM 里有笔 32-byte 的数据要用 PDMA 来传输：

- 开启 SPI 的 TX FIFO，FIFO trigger level 设 4，data length 设 8-bit
- 对应的 PDMA channel 的 SADR 设成数据在 SRAM 的起始地址，来源地址模式设成 linear increment；DADR 设成 SPI 的 DR 地址，目的地址模式设成 fixed
- Data width 设成 8-bit，block length 设 4，block count 设 8，这代表 SPI 每次发出 request，PDMA 就会依序从 SRAM 读出 4-byte 数据，连续写入 SPI DR (TX FIFO)

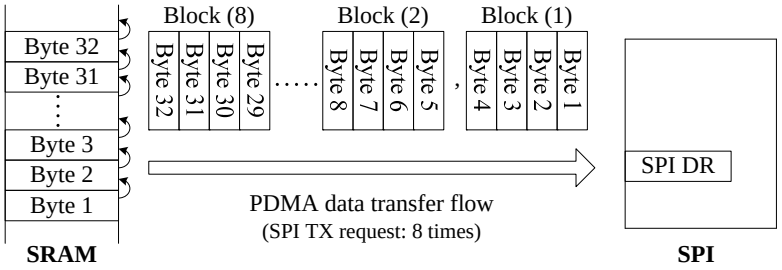

图 3　Linear Increment 和 Fixed 地址模式

使用Circular Increment与Linear Increment地址模式

以 CSIF 为例，应用程序通过 PDMA 和 CSIF，从外接的 CMOS image sensor 获取影像数据：

- 根据影像大小与 CMOS image sensor 的规格来设置 CSIF
- 对应的 PDMA channel 的 SADR 设成 CSIF FIFO Register 0 的地址，来源地址模式设成 circular incremwnt；DADR 设成影像 buffer 的起始地址，目的地址模式设成 linear increment
- Data width 设成 32-bit，block length 设成 8，block count 设成 m (根据影像数据大小来设置)，这代表每次 CSIF 发出 request，PDMA 都会依序从 CSIF FIFO register 0~7 读出 8x32-bit 的数据，写入影像 buffer 内

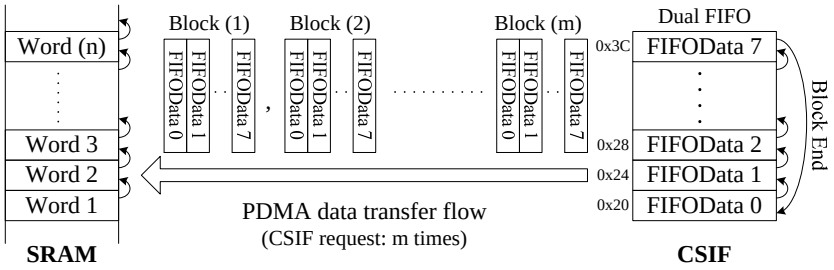

图 4　Circular increment 和 linear increment 地址模式

USB RAM数据转移

以 USB RAM 数据转移为例，使用软件触发 PDMA 从 USB RAM 读出 64-byte 数据写入 SRAM：

- 可选择任一 PDMA channel，SADR 设成 endpoint buffer 在 USB RAM 的起始地址，来源地址模式设成 linear increment & auto reload；DADR 设成 SRAM 的 buffer 起始地址，目的地址模式设成 linear increment
- Data width 设成 32-bit (USB RAM 只支持 word access)，block length 设成 4，block count 设成 4，这代表软件触发一次后，PDMA 会自动将数据分成 4 个 block，由 USB RAM 读出并写入 SRAM
- 由于开启了 auto reload，之后如果应用程序要从相同的 endpoint buffer 搬 64-byte 数据到 SRAM，只需要再启动软件触发即可；如果要更改 buffer 地址，也只需要写入新的 SADR 或 DADR 值，不需要重设 PDMA channel 的全部参数

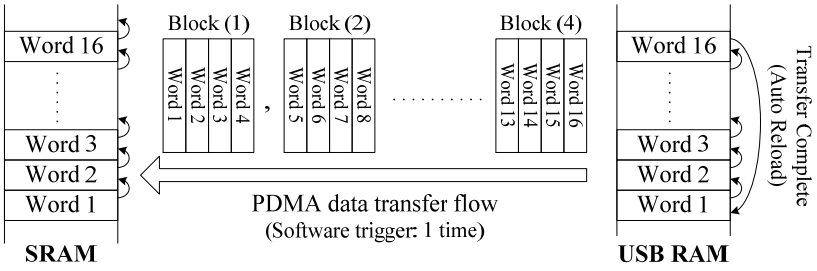

图 5　USB RAM 数据转移